Products of Ordinary Differential Operators by Evaluation and Interpolation

Alin Bostan
Algorithms Project-Team,
INRIA Paris-Rocquencourt
78153 Le Chesnay (France)
Alin.Bostan@inria.fr

Frédéric Chyzak
Algorithms Project-Team,
INRIA Paris-Rocquencourt
78153 Le Chesnay (France)
Frederic.Chyzak@inria.fr

Nicolas Le Roux
Algorithms Project-Team,
INRIA Paris-Rocquencourt
78153 Le Chesnay (France)
Nicolas.Le_Roux@inria.fr

ABSTRACT

It is known that multiplication of linear differential operators over ground fields of characteristic zero can be reduced to a constant number of matrix products. We give a new algorithm by evaluation and interpolation which is faster than the previously-known one by a constant factor, and prove that in characteristic zero, multiplication of differential operators and of matrices are computationally equivalent problems. In positive characteristic, we show that differential operators can be multiplied in nearly optimal time. Theoretical results are validated by intensive experiments.

Categories and Subject Descriptors:
I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation – *Algebraic Algorithms*

General Terms: Algorithms, Theory

Keywords: Fast algorithms, differential operators.

1. INTRODUCTION

Multiplication in polynomial algebras $\mathbb{K}[X]$ and $\mathbb{K}[X,Y]$ over a field $\mathbb{K}$ has been intensively studied in the computer-algebra literature. Since the discovery of Karatsuba's algorithm and the Fast Fourier Transform, hundreds of articles have been dedicated to theoretical and practical issues; see, e.g., [9, Ch. 8], [1], and the references therein. Not only are many other operations built upon multiplication, but often their complexity can be expressed in terms of the complexity of multiplication—whether as a constant number of multiplications or a logarithmic number of multiplications. In $\mathbb{K}[X]$, this is the case for Euclidean division, gcd and resultant computation, multipoint evaluation and interpolation, shifts, certain changes of bases, etc.

In the noncommutative setting of linear ordinary differential operators, the study is by far less advanced. The complexity of the product has been addressed only recently, by van der Hoeven in the short paper [11]: multiplication of operators over ground fields $\mathbb{K}$ of characteristic zero can be reduced by an evaluation-interpolation scheme to a constant number C of matrix multiplications with elements in $\mathbb{K}$. Work in progress [3] suggests that linear algebra is again the bottleneck for computations of GCRDs and LCLMs.

This work aims at deepening the study started in [11] for characteristic 0. We improve van der Hoeven's result along several directions: We make the constant factor C explicit in §3.2 and improve it in §4, and we prove in §3 that multiplication of matrices and of differential operators are equivalent computational problems—that is, they share the same exponent, thus answering the question left open in [11, §6, Remark 2]. As usual, those results hold for sufficiently large characteristic as well. We prolong the study to the case of (small) positive characteristic, by giving in §5 an algorithm for computing the product of two differential operators in softly quadratic complexity, that is, nearly optimally in the output size. This indicates that the equivalence result may fail to generalize to arbitrary fields.

In what follows, the field $\mathbb{K}$ has characteristic zero, unless stated otherwise. $\mathbb{K}[X]\langle\partial\rangle$ and $\mathbb{K}[X]\langle\theta\rangle$ respectively denote the associative algebras $\mathbb{K}\langle X,\partial;\partial X = X\partial+1\rangle$ and $\mathbb{K}\langle X,\theta;\theta X = X(\theta+1)\rangle$.

	vdH$_\theta$	IvdH$_\theta$	vdH	IvdH	MulWeyl
Product by blocks	37	24	96	48	12
Zeros + Strassen	20	8	47	12	8

Table 1: Number of $n\times n$ matrix products for multiplication in $\mathbb{K}[X]\langle\theta\rangle$, resp. $\mathbb{K}[X]\langle\partial\rangle$, in bidegree (n,n).

Table 1 encapsulates our improvements on the constant C. It displays the cost of linear algebra in van der Hoeven's algorithms (vdH$_\theta$, resp. vdH) and in the improved versions (IvdH$_\theta$, resp. IvdH), which are described in §3.1, resp. §4.1, and in our algorithm (MulWeyl) in §4.2. The subscript θ refers to multiplication in $\mathbb{K}[X]\langle\theta\rangle$; its absence means a product in $\mathbb{K}[X]\langle\partial\rangle$. The first row provides bounds on the number of $n\times n$ matrix products used in each algorithm for multiplying operators in $\mathbb{K}[X]\langle\partial\rangle$, resp. $\mathbb{K}[X]\langle\theta\rangle$, of degree at most n in X and in ∂, resp. θ, under the naive complexity estimate (1) below. This estimate reflects the choice of multiplying rectangular matrices by decomposing them into square blocks. The second row gives tighter bounds under the assumptions that: *(i)* any product by a zero block is discarded; *(ii)* when possible, a product of two 2×2 matrices of $n\times n$ blocks is computed as 7 block products, instead of 8, by using Strassen's algorithm [14]; *(iii)* predicted non-trivial zero blocks in the output are not computed.

Canonical form and bidegree. In the algebra $\mathbb{K}[X]\langle\partial\rangle$, resp. $\mathbb{K}[X]\langle\theta\rangle$, the commutation rule allows one to rewrite any given element into a so-called *canonical form* with X on the left of monomials and ∂, resp. θ, on the right, that is, as a linear combination of monomials $X^i\partial^j$, resp. $X^i\theta^j$, for uniquely-defined coefficients from $\mathbb{K}$. In either case, we speak of an element of bidegree (d,r), resp. at most (d,r), when the degree of its canonical form in X is d, resp. at most d, and that in ∂, resp. θ, is r, resp. at most r. With natural notation, the bidegree (d_C, r_C) of a product $C = BA$ clearly satisfies $r_C = r_A + r_B$ and $d_C \le d_A + d_B$.

The problem of computing the canonical form of the product of two elements of bidegree (d,r) from $\mathbb{K}[X]\langle\partial\rangle$, resp. from $\mathbb{K}[X]\langle\theta\rangle$, given in canonical form, is denoted $\langle d,r\rangle_\partial$, resp. $\langle d,r\rangle_\theta$.

Complexity measures. All complexity estimates are given in terms of arithmetical operations in $\mathbb{K}$, which we denote "ops." We denote by $\mathsf{C}_\theta, \mathsf{C}_\partial : \mathbb{N} \to \mathbb{N}$ two functions such that Problems $\langle n,n\rangle_\partial$ and $\langle n,n\rangle_\theta$ can be solved in $\mathsf{C}_\partial(n)$ and $\mathsf{C}_\theta(n)$, respectively. We denote by $\mathsf{M} : \mathbb{N} \to \mathbb{N}$ a function such that polynomials of degree at most n in $\mathbb{K}[X]$ can be multiplied in $\mathsf{M}(n)$ ops. Using Fast Fourier Transform algorithms, $\mathsf{M}(n)$ can be taken in $\mathcal{O}(n \log n)$ over fields with suitable roots of unity, and $\mathcal{O}(n \log n \log\log n)$ in the general case [13, 5]. We use the notation $f \in \tilde{\mathcal{O}}(g)$ for $f, g : \mathbb{N} \to \mathbb{N}$ if f is in $\mathcal{O}(g \log^m g)$ for some $m \ge 1$. For instance, $\mathsf{M}(n)$ is in $\tilde{\mathcal{O}}(n)$. The problem of multiplying an $m \times n$ matrix by an $n \times p$ matrix is written $\langle m,n,p\rangle$. We let $\mathsf{MM} : \mathbb{N}^3 \to \mathbb{N}$ be a function such that Problem $\langle m,n,p\rangle$ can be solved in $\mathsf{MM}(m,n,p)$ ops. We use the abbreviation $\mathsf{MM}(n)$ for $\mathsf{MM}(n,n,n)$. The current tightest (strict) upper bound 2.376 for ω such that $\mathsf{MM}(n) \in \mathcal{O}(n^\omega)$ is derived in [7]. For the time being, this estimate is only of theoretical relevance. Few practical algorithms with complexity better than cubic are currently known for matrix multiplication, among which Strassen's algorithm [14] with exponent $\log_2 7 \approx 2.807$ and the Pan–Kaporin algorithm [12] with exponent 2.776. For rectangular matrix multiplication, we shall use the estimate

$$\mathsf{MM}(an, bn, cn) \le abc\,\mathsf{MM}(n), \qquad \text{for } a,b,c \in \mathbb{N}, \tag{1}$$

obtained by performing the naive product of $a \times b$ by $b \times c$ matrices whose coefficients are $n \times n$ blocks.

Furthermore, we assume that $\mathsf{M}(n)$, $\mathsf{MM}(n)$, $\mathsf{C}_\partial(n)$, and $\mathsf{C}_\theta(n)$ satisfy the usual super-linearity assumption of [9, §8.3, Eq. (9)] and also that, if $\mathsf{F}(n)$ is any of these functions, then $\mathsf{F}(cn)$ belongs to $\mathcal{O}\big(\mathsf{F}(n)\big)$, for all positive constants c.

Useful complexity results. Throughout, we shall freely use several classical results on the complexity of basic polynomial operations. They are encapsulated in Lemma 1. The corresponding algorithms are found in: [8, Algorithm E] for (a); [9, Chapter 10] for (b); [10, Th. 2.4 and 2.5] for (c); and [9, Cor. 8.29] for (d).

Lemma 1 *Let $\mathbb{K}$ be an arbitrary field. Let $a \in \mathbb{K}$, let $P(X) \in \mathbb{K}[X]$ be of degree less than n and $f, g \in \mathbb{K}[X,Y]$ of degree at most d in X and n in Y. One can perform:* (a) *the Taylor shift $Q(X) := P(X+a)$;* (b) *the multipoint evaluation and interpolation of P on $a, a+1, \dots, a+n$ if the characteristic of $\mathbb{K}$ is 0 or greater than n;* (c) *the base change between the monomial and the falling factorial basis $(X)_k = X(X-1)\cdots(X-k+1)$ in $\mathcal{O}\big(\mathsf{M}(n)\log n\big)$ ops. Moreover, one computes:* (d) *the product $h = fg$ in $\mathcal{O}\big(\mathsf{M}(dn)\big)$ ops.*

2. NAIVE ALGORITHMS

In this section, we provide complexity estimates for several known algorithms for $\langle d,r\rangle_\partial$. We set

$$A = \sum_{i=0}^{r}\sum_{j=0}^{d} a_{i,j} X^j \partial^i, \quad B = \sum_{i=0}^{r}\sum_{j=0}^{d} b_{i,j} X^j \partial^i = \sum_{i=0}^{r} b_i(X)\partial^i.$$

For any $L = \sum_{i=0}^{r} l_i(X)\partial^i = \sum_{j=0}^{d} X^j l'_j(\partial)$, we define

$$\frac{dL}{dX} = \sum_{i=0}^{r} \frac{dl_i(X)}{dX}\partial^i, \qquad \frac{dL}{d\partial} = \sum_{j=0}^{d} X^j \frac{dl'_j(\partial)}{d\partial}.$$

Naive expansion. The most naive calculation of BA is by expanding each $\partial^i X^l$ in the equality

$$BA = \sum_{i=0}^{r}\sum_{j=0}^{d}\sum_{k=0}^{r}\sum_{l=0}^{d} b_{i,j} a_{k,l} X^j \left(\partial^i X^l\right)\partial^k.$$

Using Leibniz's formula $\partial^i X^l = \sum_{k=0}^{\min(i,l)} (l)_k \binom{i}{k} X^{l-k}\partial^{i-k}$ and the recurrences $(l)_{k+1} = (l)_k (l-k)$ and $\binom{i}{k+1} = \binom{i}{k}\frac{i-k}{k+1}$, the canonical form of $\partial^i X^l$ is computed in $\mathcal{O}\big(\min(i,l)\big)$ ops. This induces a complexity $\mathcal{O}\big(d^2 r^2 \min(d,r)\big)$ for computing BA. The estimate simplifies to $\mathcal{O}(n^5)$ if $d = r = n$.

Iterative schemes. Another calculation is by the formula

$$BA = \sum_{i=0}^{r} b_i(X)\left(\partial^i A\right) \tag{2}$$

and the observation that $\partial^i A$ has bidegree at most $(d, r+i)$ and is computed from $\partial^{i-1}A$ in $\mathcal{O}(dr)$ ops. by the identity

$$\partial T = T\partial + \frac{dT}{dX} \qquad \text{for } T = T(X,\partial). \tag{3}$$

Therefore, the overall complexity is $\mathcal{O}\big(\mathsf{M}(d)r^2 + dr^2\big) = \mathcal{O}\big(\mathsf{M}(d)r^2\big)$. When $d = r = n$, this is $\mathcal{O}\big(\mathsf{M}(n)n^2\big)$, and $\tilde{\mathcal{O}}\big(n^3\big)$ if FFT is used. Similar considerations based on

$$TX = XT + \frac{dT}{d\partial} \qquad \text{for } T = T(X,\partial) \tag{4}$$

provide an algorithm in $\mathcal{O}\big(d^2\,\mathsf{M}(r)\big)$, and one can always use the better algorithm by first comparing d and r.

Another formula, attributed to Takayama and used in several implementations (Takayama's Kan system [15]; Maple's $\mathsf{Ore\_algebra}$ by Chyzak [6]), is given by the (finite) sum

$$BA = \sum_{k\ge 0} \frac{1}{k!}\left(\frac{d^k B}{d\partial^k} * \frac{d^k A}{dX^k}\right), \tag{5}$$

where the products $*$ are computed formally as commutative products between canonical forms, the resulting sum being viewed as a canonical form. Each of the derivatives has bidegree at most (d,r) and the derivative at order k can be computed in $\mathcal{O}(dr)$ ops. from the one at order $k-1$. The complexity is seen to be $\mathcal{O}\big(\min(d,r)\,\mathsf{M}(dr)\big)$ ops., by Lemma 1(d). When $d = r = n$, this is $\mathcal{O}\big(n\,\mathsf{M}(n^2)\big)$, or $\tilde{\mathcal{O}}(n^3)$ using FFT; the scheme (2) is just a bit better than (5).

3. EQUIVALENCE BETWEEN PRODUCTS OF MATRICES AND OPERATORS

Let $\mathbb{K}$ be a field of characteristic zero. In [11], van der Hoeven showed that $\mathsf{C}_\theta(n)$ and $\mathsf{C}_\partial(n)$ are in $\mathcal{O}\big(\mathsf{MM}(n)\big)$. When $\omega < 3$, this improves upon the algorithms in §2.

In this section, we explain and improve this result along two directions: we make the constant factor explicit in the estimate $\mathsf{C}_\theta(n) \in \mathcal{O}\big(\mathsf{MM}(n)\big)$, and lessen it. Then, we prove that $\langle n,n,n\rangle$, $\langle n,n\rangle_\partial$, and $\langle n,n\rangle_\theta$ are equivalent computational problems, in a sense made clear below.

3.1 Product in $\mathbb{K}[X]\langle\theta\rangle$ reduces to matrix product: van der Hoeven's algorithm revisited

A differential operator A in $\mathbb{K}[X]\langle\theta\rangle$ can be viewed as a $\mathbb{K}$-endomorphism of $\mathbb{K}[X]$, mapping a polynomial f to $A(f)$. As such, it is represented, with respect to the canonical basis $(X^i)_{i\geq 0}$ of $\mathbb{K}[X]$, by an (infinite) matrix denoted M^A_∞. The submatrix of M^A_∞ consisting of its first $r \geq 1$ rows and $c \geq 1$ columns is denoted $M^A_{r,c}$.

Van der Hoeven's key observation is that an operator A of bidegree (d,r) is completely determined by the matrix $M^A := M^A_{d+r+1,r+1}$. Writing $A = \sum_{i=0}^{d}\sum_{j=0}^{r} a_{i,j}X^i\theta^j$ and using the relation $\theta^j(X^k) = k^jX^k$ yields

$$A(X^k) = \sum_{i,j} a_{i,j}k^jX^{i+k} = X^k\sum_{i=0}^{d}\tilde{A}_i(k)X^i,$$

where the polynomials $\tilde{A}_i$ are defined as $\tilde{A}_i(X) = \sum_{j=0}^{r} a_{i,j}X^j$ for all $0 \leq i \leq d$. Thus the matrix M^A has the following rectangular banded form:

$$M^A = \begin{bmatrix} \tilde{A}_0(0) & & \\ \tilde{A}_1(0) & \tilde{A}_0(1) & \\ \vdots & \tilde{A}_1(1) & \ddots & \\ \tilde{A}_d(0) & \vdots & \ddots & \tilde{A}_0(r) \\ & \tilde{A}_d(1) & & \tilde{A}_1(r) \\ & & \ddots & \vdots \\ & & & \tilde{A}_d(r) \end{bmatrix}. \qquad (6)$$

The knowledge of A is equivalent to that of all $d+1$ polynomials $\tilde{A}_i$. Each of the latter having degrees bounded by r, this is also equivalent to the data of the values $\tilde{A}_i(k)$, for $0 \leq k \leq r$ and $0 \leq i \leq d$. This is true by Lagrange interpolation. Thus, A is indeed completely determined by the $r+1$ polynomials $A(X^k)$, and also by the matrix M^A.

Now, let $A, B \in \mathbb{K}[X]\langle\theta\rangle$ and let C be BA. Then $M^C_\infty = M^B_\infty M^A_\infty$. If A, B, and C have bidegrees (d_A, r_A), (d_B, r_B), and (d_C, r_C), then the previous discussion implies the following "finite version" of this matrix equality:

$$M^C = M^B_{d_C+r_C+1,d_A+r_C+1}M^A_{d_A+r_C+1,r_C+1}, \qquad (7)$$

which is the basis of the algorithm in [11], described below.

Putting all these considerations together leads to Algorithm Mul_θ in Fig. 1 and proves the following proposition.

Proposition 1 *Algorithm Mul_θ in Fig. 1 reduces the computation of the product $C = BA$ to the following tasks:*

(T1) *d_A+1 evaluations in degrees $\leq r_A$ on $0,1,\dots,r_C$;*

(T2) *d_B+1 evaluations in degrees $\leq r_B$ on $0,1,\dots,d_A+r_C$;*

(T3) *d_C+1 interpolations in degrees $\leq r_C$ on $0,1,\dots,r_C$;*

(T4) *an instance of $\langle d_C+r_C+1, d_A+r_C+1, r_C+1\rangle$.*

Proof. Eq. (6) shows that Step 1 in Algorithm Mul_θ is performed by the evaluation Tasks (T1–T2). Similarly, the interpolation Task (T3) performs Step 3. Finally, the product in Step 2 is computed by (T4). ☐

```
Mul_θ(B, A)

Input: $A, B \in \mathbb{K}[X]\langle\theta\rangle$.
Output: their product $C = BA$.
1. Compute the $\tilde{A}_i$'s and $\tilde{B}_i$'s from $A$ and $B$, then the
   matrices $M^B_{d_C+r_C+1,d_A+r_C+1}$ and $M^A_{d_A+r_C+1,r_C+1}$.
2. Compute $M^C$ by Eq. (7).
3. Compute the $\tilde{C}_i$'s from $M^C$, then recover $C$.
```

Figure 1: Product of differential operators in θ.

We stress that the evaluation-interpolation scheme used in Algorithm Mul_θ requires that the interpolation points $0,1,\dots,r_C$ be mutually distinct. Thus, this scheme would not have worked over a field of small characteristic, but would have remained valid in large enough characteristic.

In the original article [11], Tasks (T1–T3) are performed by matrix multiplications, as explained in the next lemma.

Lemma 2 *Let $d,r,s \in \mathbb{N}$ and let $a_0,\dots,a_s$ be distinct points in $\mathbb{K}$. Evaluating $d+1$ polynomials of degree r on the a_i's reduces to an instance of $\langle s+1,r+1,d+1\rangle$ plus $\mathcal{O}(sr)$ ops. Interpolating $d+1$ polynomials of degree s on the a_i's amounts to an instance of $\langle s+1,s+1,d+1\rangle$ plus $\mathcal{O}(s^2)$ ops.*

Proof. The omitted proof is based on grouping multiplications by Vandermonde matrices into a single product. ☐

Using Lemma 2, one immediately deduces the cost of van der Hoeven's algorithm "à la lettre" (vdH_θ); the following enumeration displays only the dominating costs, quadratic estimates like $O(r_Cr_A)$ being intentionally neglected:

1. $\mathsf{MM}(r_C+1, r_A+1, d_A+1)$ for (T1);
2. $\mathsf{MM}(d_A+r_C+1, r_B+1, d_B+1)$ for (T2);
3. $\mathsf{MM}(r_C+1, r_C+1, d_C+1)$ for (T3);
4. $\mathsf{MM}(d_C+r_C+1, d_A+r_C+1, r_C+1)$ for (T4).

Notice that the last step dominates the cost.

For Problem $\langle n,n\rangle_\theta$ which is studied in [11], applying the estimate (1) leads to the number $2+3+2\cdot2\cdot2+4\cdot3\cdot2 = 37$ of $n\times n$ block multiplications given in column vdH_θ of Table 1. This estimate is however pessimistic and can be reduced to 20: Strassen's formula reduces the 8 block products in Task (T3) to 7; the band structure of the matrices in Task (T4) reduces 24 to only 8 products of non-zero blocks.

A first improvement. Algorithm vdH_θ can be improved by making use of fast multipoint evaluation and interpolation of Lemma 1(b) to perform Steps 1 and 3 of Algorithm Mul_θ in Fig. 1. This remark will be crucial in our proof of equivalence in §3.2. We arrive at the following complexity estimates:

1. $\mathcal{O}\big(d_A \,\mathsf{M}(r_C)\log r_C\big)$ for (T1);
2. $\mathcal{O}\big(d_B \,\mathsf{M}(d_A+r_C)\log(d_A+r_C)\big)$ for (T2);
3. $\mathcal{O}\big(d_C \,\mathsf{M}(r_C)\log r_C\big)$ for (T3).

Assuming FFT is available for polynomial multiplication, the cumulated cost of Tasks (T1–T3) drops to

$$\tilde{\mathcal{O}}\big(d_A r_C + d_B(d_A+r_C) + d_C r_C\big) \in \tilde{\mathcal{O}}\big(d_C r_C + d_A d_B\big).$$

This cost is nearly optimal, since it is almost linear in the number of non-zero elements of the matrices involved in Eq. (7). In the particular case of problem $\langle n,n\rangle_\theta$, we obtain the numbers 24 and 8 of column IvdH_θ in Table 1.

3.2 Matrix multiplication reduces to product in $\mathbb{K}[X]\langle\theta\rangle$

In summary, the results of the previous section show that $\mathsf{C}_\theta(n) \in \mathcal{O}\big(\mathsf{MM}(n)\big)$. Here we prove the converse statement, by proceeding in two steps. First, Lemma 3 shows that the multiplication, whose complexity is denoted $\mathsf{T}(n)$, of two lower-triangular matrices of size $n\times n$ reduces to the product of two operators of bidegree at most (n,n) in (X,θ). Secondly, Lemma 4 proves that multiplying two arbitrary matrices amounts to a constant number of products of lower-triangular matrices.

Lemma 3 $\mathsf{T}(n) \in \mathsf{C}_\theta(n) + \mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$.

Proof. Let L_1, L_2 be two $(n+1)\times(n+1)$ lower-triangular matrices. Denote $(t_{i,j})$ and $(s_{i,j})$ their coefficients, with $0\le i,j\le n$. Let $\tilde{B}_\ell(X)$ and $\tilde{A}_\ell(X)$ be the (unique) polynomials in $\mathbb{K}[X]$ of degree at most $n-\ell$ that interpolate the elements of the ℓth lower diagonal of L_1, resp. L_2, on the set $\{0,1,\dots,n-\ell\}$. Specifically, for $0\le \ell, j\le n$ with $\ell+j\le n$, we have $t_{\ell+j,j} = \tilde{B}_\ell(j)$ and $s_{\ell+j,j} = \tilde{A}_\ell(j)$. Using fast interpolation, the computation of the polynomials $\tilde{B}_\ell(X)$ and $\tilde{A}_\ell(X)$, for $0\le\ell\le n$, is done in $\mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$ ops. Define $A = \sum_{\ell=0}^n \sum_{j=0}^{n-\ell} a_{\ell,j} X^\ell \theta^j$ and $B = \sum_{\ell=0}^n \sum_{j=0}^{n-\ell} b_{\ell,j} X^\ell\theta^j$ from the coefficients in $\tilde{A}_\ell(X) = \sum_{j=0}^{n-\ell} a_{\ell,j}X^j$ and $\tilde{B}_\ell(X) = \sum_{j=0}^{n-\ell} b_{\ell,j}X^j$. Let $C = BA$. Then, L_1 and L_2 are seen to be top-left blocks of M^B and M^A, and Eq. (7) with A replaced by C shows that the top-left $(n+1)\times(n+1)$ submatrix of M^C is the lower-triangular matrix L_1L_2. This submatrix is computed starting from the coefficients of C using $\mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$ ops., by fast multipoint evaluation. $\square$

Lemma 4 $\mathsf{MM}(n) \in \mathcal{O}\big(\mathsf{T}(n)\big)$.

Proof. Let M, N be $n\times n$ matrices. The identity

$$\begin{bmatrix} I_n & 0 & 0 \\ M & I_n & 0 \\ 0 & N & I_n \end{bmatrix}^2 = \begin{bmatrix} I_n & 0 & 0 \\ 2M & I_n & 0 \\ NM & 2N & I_n \end{bmatrix}$$

shows that $\mathsf{MM}(\lceil n/3\rceil) \le \mathsf{T}(n) \in \mathcal{O}(\mathsf{T}(n))$ and the conclusion follows from the growth hypotheses on MM. $\square$

Lemmas 3 and 4 imply the main result of this section.

Theorem 1 *There exists a constant $K>0$ such that*

$$\mathsf{MM}(n) \le K\big(\mathsf{C}_\theta(n) + n\,\mathsf{M}(n)\log n\big).$$

3.3 Equivalence between product in $\mathbb{K}[X]\langle\partial\rangle$ and in $\mathbb{K}[X]\langle\theta\rangle$

Relax $\mathbb{K}$ to be a field of arbitrary characteristic. Any operator $A = \sum_{i=0}^r \alpha_i\theta^i$ in $\mathbb{K}[X]\langle\theta\rangle$ with coefficients α_i of degree at most d can be expressed in the algebra $\mathbb{K}[X]\langle\partial\rangle$ as $A = \sum_{i=0}^r a_i\partial^i$, with coefficients a_i of degree at most $d+r$.

As indicated in the proof of [4, Cor. 2], performing the conversion from the representation in θ to the representation in ∂ amounts to multiplying a Stirling matrix S of size $r+1$ by an $(r+1)\times(d+1)$ matrix containing the coefficients of the α_i's. This matrix product can be decomposed into $d+1$ matrix-vector products of the form $w = Sv$. The coefficients of the vector w represent the coefficients of the polynomial $\sum_i v_i X^i$ in the falling factorial basis $(X)_k$. As Lemma 1(c) holds for any characteristic, w can be computed using $\mathcal{O}\big(\mathsf{M}(r)\log r\big)$ ops. To summarize, the coefficients a_i can be computed from the α_i's in $\mathcal{O}\big(d\,\mathsf{M}(r)\log r\big)$ ops.

Conversely, let $B = \sum_{i=0}^r b_i\partial^i$ be in $\mathbb{K}[X]\langle\partial\rangle$. It can be written in the algebra $\mathbb{K}[X,X^{-1}]\langle\theta\rangle$ of differential operators in θ with Laurent polynomial coefficients as follows: $B = \sum_{i=0}^r \beta_i\theta^i$. If the b_i's have degrees bounded by d, then the β_i's have degrees at most d and valuation at least $-r$ in X. A discussion similar to above shows that the computation of the coefficients β_i from the coefficients b_i amounts to multiplying the inverse of the Stirling matrix by an $(r+1)\times(d+r+1)$ matrix. This matrix product can be decomposed into $d+r+1$ matrix-vector products by S^{-1}; this amounts to expanding in the monomial basis $d+r+1$ polynomials of degree at most r given in the falling factorial basis. Thus, the conversion can be done in $\mathcal{O}\big((d+r)\,\mathsf{M}(r)\log r\big)$ ops.

We encapsulate this discussion into the following result, which proves that $\langle n,n\rangle_\partial$ and $\langle n,n\rangle_\theta$ are computationally equivalent, up to $\tilde{\mathcal{O}}(n^2)$ terms, in any characteristic.

Theorem 2 *There exist a constant $C>0$ such that*

$$\mathsf{C}_\theta(n) \le C\big(\mathsf{C}_\partial(n) + n\,\mathsf{M}(n)\log n\big),$$
$$\mathsf{C}_\partial(n) \le C\big(\mathsf{C}_\theta(n) + n\,\mathsf{M}(n)\log n\big),$$

over fields of any characteristic.

Proof. Let A_1, A_2 be of bidegree (n,n) in $\mathbb{K}[X]\langle\theta\rangle$. By the previous discussion, converting them into $\mathbb{K}[X]\langle\partial\rangle$ has cost $\mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$. Both A_1, A_2 have bidegrees at most $(2n,n)$ in (X,∂) and can thus be multiplied using $\mathsf{C}_\partial(2n)\in\mathcal{O}\big(\mathsf{C}_\partial(n)\big)$ ops. Converting the result back into $\mathbb{K}[X]\langle\theta\rangle$ costs $\mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$ ops. This proves the first inequality.

Let now B_1 and B_2 be of bidegree (n,n) in $\mathbb{K}[X]\langle\partial\rangle$. Their conversion in $\mathbb{K}[X,X^{-1}]\langle\theta\rangle$ can be performed using $\mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$ ops. and produces two operators C_1 and C_2 in $\mathbb{K}[X]\langle\theta\rangle$, of bidegrees at most $(2n,n)$ in (X,θ) such that $B_1 = X^{-n}C_1(X,\theta)$ and $B_2 = X^{-n}C_2(X,\theta)$. Using the commutation rule $C_2(X,\theta)X^{-n} = X^{-n}C_2(X,\theta-n)$, we deduce the equality $B_2B_1 = X^{-2n}C_2(X,\theta-n)C_1(X,\theta)$. Writing $C_2(X,\theta) = \sum_{j=0}^n X^j c_j'(\theta)$ shows that computing the coefficients of $C_2(X,\theta-n)$ amounts to $n+1$ polynomial shifts in $\mathbb{K}[\theta]$ in degree at most n. Each of these shifts can be computed in $\mathcal{O}\big(\mathsf{M}(n)\log n\big)$ ops., using Lemma 1(a). Conversion of B_2B_1 back into $\mathbb{K}[X]\langle\partial\rangle$ has the same cost. $\square$

4. BETTER CONSTANTS IN $\mathbb{K}[X]\langle\partial\rangle$

In §4.1, we revisit van der Hoeven's algorithm for $\langle n,n\rangle_\partial$ and exhibit the constant factor in its $\mathcal{O}\big(\mathsf{MM}(n)\big)$ cost. Then, we propose in §4.2 a new algorithm with a better constant.

4.1 Multiplication in $\mathbb{K}[X]\langle\partial\rangle$: van der Hoeven's algorithm revisited

Van der Hoeven's algorithm for computing products in $\mathbb{K}[X]\langle\partial\rangle$ is based on the fact that his algorithm for products in $\mathbb{K}[X]\langle\theta\rangle$ can be adapted to operators with Laurent polynomials coefficients. Indeed, to any $L \in \mathbb{K}[X, X^{-1}]\langle\theta\rangle$ of the form $L = \sum_{i=-v}^{d}\sum_{j=0}^{r} \ell_{i,j}X^i\theta^j$ is associated an infinite matrix representing the $\mathbb{K}$-linear map of multiplication by L from $\mathbb{K}[X]$ to $X^{-v}\mathbb{K}[X]$. Its $(v+d+r+1)\times(r+1)$-submatrix $M^L_{0,r}$ (defined shortly) is banded and it uniquely determines the operator L, as in the case of polynomial coefficients.

To be precise, for two integers $\alpha \le \beta$ we denote by $M^L_{\alpha,\beta}$ the $(v+d+\beta-\alpha+1)\times(\beta-\alpha+1)$ matrix whose $(\gamma-\alpha+1)$-th column, for $\alpha \le \gamma \le \beta$, contains the coefficients of $L(X^\gamma)$ on $X^{-v+\alpha}, \dots, X^{d+\beta}$. The matrix $M^L_{\alpha,\beta}$ has a banded form and contains on its diagonals the evaluations on the points $\alpha, \dots, \beta$ of the polynomials $\tilde{L}_{-v}, \dots, \tilde{L}_d$ defined by $\tilde{L}_i(X) = \sum_{j=0}^{r} \ell_{i,j}X^j$ for all $-v \le i \le d$.

Let A, B have valuations $-v_A, -v_B$ and degrees d_A, d_B with respect to X, and degrees r_A, r_B in θ. If $C = BA$ in $\mathbb{K}[X, X^{-1}]\langle\theta\rangle$, then the following equality, analogous to Eq. (7), holds:

$$M^C_{0,r_C} = M^B_{-v_A,d_A+r_C} M^A_{0,r_C}. \tag{8}$$

Likewise, the product of operators in $\mathbb{K}[X, X^{-1}]\langle\theta\rangle$ reduces to some evaluation and interpolation tasks (in order to convert between operators and matrices) and to the main matrix-multiplication task (8), which is an instance of $\langle v_C + d_C + r_C + 1, v_A + d_A + r_C + 1, r_C + 1\rangle$.

The algorithm for multiplication in $\mathbb{K}[X]\langle\partial\rangle$ based on multiplication in $\mathbb{K}[X, X^{-1}]\langle\theta\rangle$ is described in Fig. 2 below.

```
                    Mul_∂(B, A)

Input: $A, B \in \mathbb{K}[X]\langle\partial\rangle$.
Output: their product $C = BA$.

1. Convert $A, B$ in $\mathbb{K}[X, X^{-1}]\langle\theta\rangle$.
2. Compute the product $C = BA$ in $\mathbb{K}[X, X^{-1}]\langle\theta\rangle$:
     2.1 From $A$ and $B$, compute the matrices
           $M^B_{-v_A,d_A+r_C}$ and $M^A_{0,r_C}$.
     2.2 Compute the matrix $M^C_{0,r_C}$ using Eq. (8).
     2.3 Recover $C$ from $M^C_{0,r_C}$.
3. Convert $C$ in $\mathbb{K}[X]\langle\partial\rangle$ and return it.
```

Figure 2: Product of differential operators in ∂.

In what follows, we treat in more detail the main case of interest, $\langle n, n\rangle_\partial$, as solved by Algorithm Mul_∂ in Fig. 2. Van der Hoeven suggests to perform Steps 1 and 3 using matrix multiplications by Stirling matrices and their inverses [11, §5.1, Eqs. (12–13)] and Steps 2.1 and 2.3 using matrix multiplications by Vandermonde matrices and their inverses [11, §2 and §4]. The elements of all the needed Stirling and Vandermonde matrices (and their inverses) can be computed using $\mathcal{O}(n^2)$ ops. A careful inspection of the matrix sizes involved in Algorithm Mul_∂ shows that:

1. Step 1 reduces to 2 instances of $\langle 2n+1, n+1, n+1\rangle$;
2. Step 3 reduces to an instance of $\langle 4n+1, 2n+1, 2n+1\rangle$;
3. Step 2.1 reduces to an instance of $\langle 2n+1, n+1, 4n+1\rangle$ and an instance of $\langle 2n+1, n+1, 2n+1\rangle$;
4. Step 2.3 reduces to an instance of $\langle 4n+1, 2n+1, 2n+1\rangle$;
5. Step 2.2 reduces to an instance of $\langle 6n+1, 4n+1, 2n+1\rangle$.

This variant of the algorithm is what we call vdH. Using again the estimate (1) yields the constant 96 in Table 1.

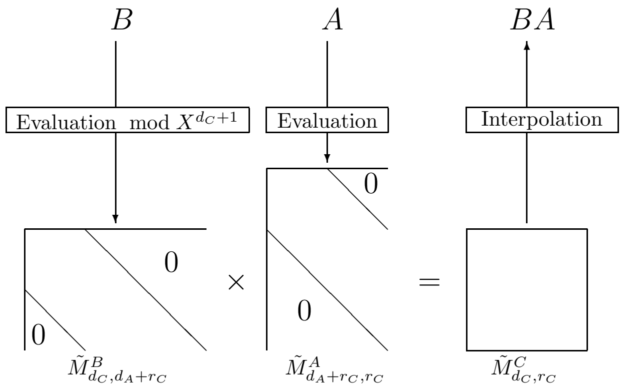
Several Improvements. A first improvement on vdH is to use fast multipoint evaluation and interpolation for Steps 2.1 and 2.3. A second improvement concerns conversions back and forth between operators in $\mathbb{K}[X, X^{-1}]\langle\partial\rangle$ and in $\mathbb{K}[X, X^{-1}]\langle\theta\rangle$ (Steps 1 and 3). Instead of using matrix products by Stirling matrices and their inverses, one can apply Lemma 1(c), as explained in §3.3. Both improvements in conjunction with FFT lessen the cost of Steps 1, 2.1, 2.3, and 3 to a negligible $\tilde{\mathcal{O}}(n^2)$. We call this improved algorithm IvdH. Using (1) yields the constant 48 in column IvdH in Table 1. The constants 47 and 12 on the last row of the table are more technical and will be proved in [2]. They rely on observing that the output of IvdH requires partial calculation of (8), reducing to an instance of $\langle 4n+1, 3n+1, 2n+1\rangle$.

4.2 A new, direct evaluation-interpolation algorithm

Let A and B be in $\mathbb{K}[X]\langle\partial\rangle$ with respective bidegrees (d_A, r_A) and (d_B, r_B). We give here an evaluation-interpolation algorithm for computing $C = BA$ which essentially reduces to $\langle d_C + 1, d_A + r_C + 1, r_C + 1\rangle$ for those bidegrees.

To achieve this, we interpret again a differential operator P in $\mathbb{K}[X]\langle\partial\rangle$ as a $\mathbb{K}$-endomorphism of $\mathbb{K}[X]$, and represent it in the canonical basis $(X^i)_{i\ge 0}$ by an (infinite) matrix denoted $\tilde{M}^P_\infty$. The submatrix of $\tilde{M}^P_\infty$ consisting of its first $r+1 \ge 1$ rows and $c+1 \ge 1$ columns is denoted $\tilde{M}^P_{r,c}$.

Then, much like Algorithm Mul_θ in §3.1, our new algorithm $\mathsf{MulWeyl}$ in Fig. 4 relies on the key observation that an operator $P \in \mathbb{K}[X]\langle\partial\rangle$ of bidegree (d, r) is uniquely determined by the submatrix $\tilde{M}^P_{d,r}$ of $\tilde{M}^P_\infty$. This key fact is proved in Theorem 4 below. The principle of the algorithm is given in Fig. 3, where evaluation and interpolation are performed by truncated-series products. In the case of $\langle n, n\rangle_\partial$, the corresponding matrices become $\tilde{M}^B_{2n,3n}$, $\tilde{M}^A_{3n,2n}$, and $\tilde{M}^C_{2n,2n}$.



B
A
BA
Evaluation mod X^{d_C+1}
Evaluation
Interpolation
0
0
0
0
$\times$
$=$
$\tilde{M}^B_{d_C,d_A+r_C}$
$\tilde{M}^A_{d_A+r_C,r_C}$
$\tilde{M}^C_{d_C,r_C}$

Figure 3: Evaluation-Interpolation w.r.t. ∂.

Theorem 3 *Algorithm MulWeyl is correct and uses* $\mathsf{MM}(d_C+1, d_A+r_C+1, r_C+1) + \tilde{\mathcal{O}}\big((d_C+r_C)^2\big)$ *ops.*

```
MulWeyl$(B, A)$

Input: $A, B \in \mathbb{K}[X]\langle\partial\rangle$.
Output: their product $C = BA$.

1. Construct the matrices $\tilde{M}^A_{d_A+r_C,r_C}$ and $\tilde{M}^B_{d_C,d_A+r_C}$.
2. Compute the product $\tilde{M}^B_{d_C,d_A+r_C}\tilde{M}^A_{d_A+r_C,r_C}$.
3. Recover $C$ from the product in Step 2.
```

Figure 4: Product of differential operators in ∂.

Proof. By the definition of the matrix $\tilde{M}^P_{r,c}$, the matrices constructed in Step 1 are associated to the linear map which sends $f \in \mathbb{K}[X]_{\le r_C}$ to $A(f)$ in $\mathbb{K}[X]_{\le d_A+r_C}$ and to the linear map which sends $f \in \mathbb{K}[X]_{\le d_A+r_C}$ to $B(f) \bmod X^{d_C+1}$ in $\mathbb{K}[X]_{\le d_C}$. Therefore, the product at Step 2 delivers the $(BA)(X^i) \bmod X^{d_C+1}$, $0 \le i \le r_C$. The $(d_C+1)\times(r_C+1)$ matrix computed is thus equal to $\tilde{M}^C_{d_C,r_C}$. This is summarized in the identity $\tilde{M}^B_{d_C,d_A+r_C}\tilde{M}^A_{d_A+r_C,r_C} = \tilde{M}^C_{d_C,r_C}$, in which the structure of zeros is given in Fig. 3. The interpolation of Step 3 relies on Theorem 4 below, which shows that $C = BA$ is fully and uniquely determined by $\tilde{M}^C_{d_C,r_C}$. This terminates the correctness proof. The claimed complexity derives immediately from Propositions 2 and 3 that are proved in the next subsections. $\square$

4.2.1 Interpolation theorem

We now state the main interpolation result, which we prove after recalling a useful filtration on $W = \mathbb{K}[X]\langle\partial\rangle$.

Theorem 4 *For $d, r \in \mathbb{N}$, let $W_{d,r}$ denote its $\mathbb{K}$-subspace*

$$W_{d,r} = \{\, P \in W \ : \ \deg_X(P) \le d,\ \deg_\partial(P) \le r \,\}.$$

Then, an isomorphism is given by the $\mathbb{K}$-linear map

$$\mathrm{EvOp}_{d,r} \ : \ \begin{array}{ccc} W_{d,r} & \to & \mathbb{K}^{(d+1)\times(r+1)} \\ P & \mapsto & \tilde{M}^P_{d,r} \end{array}.$$

In order to prove Theorem 4, we use the filtration on W defined by the weights 1 on X and -1 on ∂. The decomposition into homogeneous components of any $P \in W_{d,r}$ only involves weights between $-r$ and d. It actually admits a special form, to be exploited later, which is described now.

Lemma 5 *The homogeneous decomposition of $P \in W_{d,r}$ is*

$$P = \sum_{i=1}^{r} \ell_{-i}(X\partial)\partial^i + \sum_{i=0}^{d} X^i \ell_i(X\partial),$$

where the ℓ_i's and ℓ_{-i}'s are polynomials of degree at most $\mu_i := \min(d-i, r)$ and $\mu_{-i} := \min(r-i, d)$, respectively.

Proof. Let P be $\sum_{i,j} p_{i,j} X^j \partial^i$. Then P decomposes as the sum of $\sum_{i>j} p_{i,j}(X^j\partial^j)\partial^{i-j}$ and of $\sum_{i\le j} X^{j-i} p_{i,j}(X^i\partial^i)$. Here $p_{i,j}$ is zero if $i > r$ or if $j > d$, therefore P is equal to

$$\sum_{s=1}^{r}\left(\sum_{j=0}^{\mu_{-s}} p_{j+s,j} X^j\partial^j\right)\partial^s + \sum_{t=0}^{d} X^t \left(\sum_{i=0}^{\mu_t} p_{i,t+i} X^i\partial^i\right). \quad (9)$$

Since any $X^i\partial^i$ can be written as a polynomial of degree i in $X\partial$, the conclusion follows by expressing each parenthesis in (9) as a polynomial in $X\partial$. $\square$

Proof of Th. 4. Since $\dim_{\mathbb{K}} W_{d,r}$ is $\dim_{\mathbb{K}} \mathbb{K}^{(d+1)\times(r+1)}$, it suffices to show that $\mathrm{EvOp}_{d,r}$ is injective. Let P in $W_{d,r}$ be such that $\tilde{M}^P_{d,r} = 0$, or equivalently $P(X^k) \bmod X^{d+1} = 0$ for all $0 \le k \le r$. The decomposition of $P \in W_{d,r}$ in Lemma 5 enables one to evaluate it easily at X^k for $k \le r$:

$$P(X^k) = \sum_{i=1}^{k} \frac{k!}{(k-i)!}\ell_{-i}(k-i)X^{k-i} + \sum_{i=0}^{d} \ell_i(k) X^{k+i}. \quad (10)$$

Since $P(X^k) \bmod X^{d+1} = 0$ for $k \le r$, Eq. (10) implies:

- $\ell_i(k) = 0$ if $0 \le i \le d$, $0 \le k \le r$, and $k + i \le d$,
- $\ell_{-i}(k-i) = 0$ if $1 \le i \le k$, $0 \le k \le r$, and $k - i \le d$.

These equalities show that $\ell_i(0), \dots, \ell_i\big(\min(d-i, r)\big)$ are zero for $0 \le i \le d$ and that $\ell_{-i}(0), \dots, \ell_{-i}\big(\min(r-i, d)\big)$ are zero for $1 \le i \le r$. Finally, Lagrange interpolation and the degree bounds in Lemma 5 imply that all the polynomials ℓ_i and ℓ_{-i} are identically zero. Thus, P is 0. $\square$

A direct use of the ideas of this subsection would now end the proof of Theorem 3; the corresponding algorithm would first compute the polynomials ℓ_i and ℓ_{-i}, before evaluating them on $0, 1, \dots$. By the following next two subsections, we shall propose a better solution, avoiding a logarithmic factor and hiding a smaller constant in the $\tilde{\mathcal{O}}(\cdot)$ term.

4.2.2 Evaluation step

Here we focus on Step 1 of Algorithm MulWeyl, which is an instance of the task of computing the matrix $\tilde{M}^P_{m,n}$ for given $P = \sum_{i=0}^{r}\sum_{j=0}^{d} p_{i,j}X^j\partial^i$ in W and integers $m \ge d, n \ge r$. The announced better approach makes use of Algorithm Eval in Fig. 5, which is based on the following observation: Let $0 \le k \le n$. Then we have the identities

$$P\left(X^k\right) = \sum_{i=0}^{\min(r,k)}\sum_{j=0}^{d} p_{i,j}\frac{k!}{(k-i)!}X^{k+j-i}$$

$$= k!\,X^k \left(\sum_{\ell=-\min(r,k)}^{d}\left(\sum_{i=\max(0,-\ell)}^{\min(r,d-\ell,k)} \frac{p_{i,i+\ell}}{(k-i)!}\right)X^\ell\right).$$

Therefore, for $-r \le \ell \le d$ and $0 \le k \le n$, the coefficient $(S_\ell)_k$ of X^k in the polynomial product

$$S_\ell = \left(\sum_{i=\max(0,-\ell)}^{\min(r,d-\ell)} p_{i,i+\ell}X^i\right)\left(\sum_{j=0}^{n}\frac{X^j}{j!}\right) \quad (11)$$

gives the coefficient of X^ℓ in $\left(k!\,X^k\right)^{-1}P(X^k)$. Thus the coefficients $(S_\ell)_k$ for $\max(0,-\ell) \le k \le \min(m-\ell, n)$ of S_ℓ are, up to factorials, the coefficients on a certain diagonal of the matrix $\tilde{M}^P_{m,n}$, the other diagonals of $\tilde{M}^P_{m,n}$ being zero.

Proposition 2 *Algorithm Eval computes $\tilde{M}^P_{m,n}$ in $\mathsf{M}(mn) + \mathcal{O}(mn)$ ops.*

Proof. The series $\exp(X) \bmod X^{n+1}$ and the factorials $1, \dots, n!$ are computed by recurrence relations in $\mathcal{O}(n)$ ops. The computation of S_ℓ can be done in $\mathsf{M}(s_\ell)$ for the size s_ℓ of the corresponding diagonal of $\tilde{M}^P_{m,n}$. Summing over ℓ and appealing to properties of M leads to $\sum_\ell \mathsf{M}(s_\ell) \le \mathsf{M}\big(\sum_\ell s_\ell\big) \le \mathsf{M}(mn) + \mathcal{O}(mn)$, then to the announced complexity. $\square$

```
Eval(P)

Input: $P \in W_{d,r}$, $m \geq d$, $n \geq r$.
Output: $\tilde{M}^P_{m,n}$.

1. For each $-r \leq \ell \leq d$, compute $S_\ell \bmod X^{\min(m-\ell,n)+1}$
     by using Eq. (11).
2. Initialize $M$ to be an $(m+1) \times (n+1)$ zero matrix.
3. For $-r \leq \ell \leq d$ and $\max(0,-\ell) \leq k \leq \min(m-\ell, n)$,
     $M_{k+\ell,k} := k!\,(S_\ell)_k$.
```

Figure 5: Evaluation in $\mathbb{K}[X]\langle\partial\rangle$.

4.2.3 *Interpolation step*

Given a $(d+1) \times (r+1)$ matrix M, Step 3 of Algorithm MulWeyl computes the only operator $P \in W_{d,r}$ satisfying $\tilde{M}^P_{d,r} = M$. This is done by inverting Eq. (11). The resulting algorithm is described in Fig. 6. A similar analysis to that of algorithm Eval leads to the estimate in Proposition 3.

```
Interpol(M)

Input: $M \in \mathbb{K}^{(d+1)\times(r+1)}$.
Output: $P \in W_{d,r}$ such that $\tilde{M}^P_{d,r} = M$.

1. Divide the $k$th column of $M$ by $k!$.
2. For each $-r \leq \ell \leq d$, compute the product $T_\ell =$
     $\left(\sum_{k=\max(0,-\ell)}^{\min(d-\ell,r)} M_{k+\ell,k} X^k\right) \exp(-X) \bmod X^{\min(d-\ell,r)+1}$.
3. Return $\sum_{i=0}^{r} \sum_{\ell=-\min(i,r)}^{\min(d-i,r)} (T_\ell)_i X^{\ell+i} \partial^i$.
```

Figure 6: Interpolation in $\mathbb{K}[X]\langle\partial\rangle$.

Proposition 3 *Interpol computes P in $\mathsf{M}(dr) + \mathcal{O}(dr)$ ops.*

4.3 Comparison of algorithms for $\langle n,n\rangle_\partial$

Algorithms Mul_∂ in Fig. 2 and MulWeyl in Fig. 4 follow the same scheme: construction of evaluation matrices associated to A and B; product of these matrices; reconstruction of C by interpolation from it. But they differ in the way to do this, and MulWeyl can be viewed as an improvement on Mul_∂: the matrices computed by MulWeyl are submatrices of M^B and M^A in Algorithm Mul_∂, as will be proved in [2]. Taking accurate sizes into account for $\langle n,n\rangle_\partial$, the dominant matrix-product problem drops from $\langle 6n+1, 4n+1, 2n+1\rangle$ to $\langle 2n+1, 3n+1, 2n+1\rangle$. Estimate (1) yields the number 12 in the last column of Table 1. Observing that the product at Step 2 of MulWeyl reduces to one instance of $\langle 2n+1, 2n+1, 2n+1\rangle$ and one of $\langle n,n,n\rangle$, and appealing to Strassen's formula again, we obtain $7+1=8$ block products, as given on the last row of Table 1.

5. PRODUCT IN CHARACTERISTIC >0

As already pointed out, the evaluation-interpolation algorithms of Sections 3 and 4 remain valid when the characteristic p of $\mathbb{K}$ is positive and sufficiently large, but they fail to work in small characteristic. For instance, MulWeyl solves Problem $\langle n,n\rangle_\partial$ for characteristic $p > 3n$.

In this section, we provide an algorithm of different nature which proves that, in characteristic p, the product of two operators of bidegree (n,n) either in $\mathbb{K}[X]\langle\theta\rangle$ or in $\mathbb{K}[X]\langle\partial\rangle$ can be computed in $\tilde{\mathcal{O}}(pn^2)$ ops. For small p, this result is nearly optimal, since it is softly linear in the output size.

Up to $\tilde{\mathcal{O}}(n^2)$ additional ops., multiplication in $\mathbb{K}[X]\langle\partial\rangle$ can be reduced to multiplication in $\mathbb{K}[X]\langle\theta\rangle$, as explained in §3.3. Thus, we focus on Problem $\langle n,n\rangle_\theta$.

Our algorithm $\mathsf{Mul}_{\theta,p}$ for multiplication in $\mathbb{K}[X]\langle\theta\rangle$ is given in Fig. 7. It is based on the key fact that θ and X^p commute in characteristic p. This is used in Step 2, which reduces the product in $\mathbb{K}[X]\langle\theta\rangle$ to several products in the commutative polynomial ring $\mathbb{K}[X^p,\theta]$.

```
Mul_{θ,p}(B, A)

Input: $A, B \in \mathbb{K}[X]\langle\theta\rangle$, with $\mathrm{char}(\mathbb{K}) = p > 0$.
Output: their product $C = BA$.

1. Rewrite $A$ and $B$ as $A = \sum_{v=0}^{p-1} A_v(X^p,\theta) X^v$
     and $B = \sum_{u=0}^{p-1} X^u B_u(X^p,\theta)$.
2. Compute the commutative bivariate products
     $C_{u,v} = B_u A_v$, for $0 \leq u, v < p$.
3. Write $\sum_{u,v=0}^{p-1} X^u C_{u,v}(X^p,\theta) X^v$ in canonical form;
     return it.
```

Figure 7: Product of differential operators in θ over a field of positive characteristic.

We now describe proper algorithmic choices that perform each step of $\mathsf{Mul}_{\theta,p}$ in nearly optimal complexity.

Step 1 first rewrites A as $\sum_{v=0}^{p-1} X^v \tilde{A}_v(X^p,\theta)$ and B as $\sum_{u=0}^{p-1} X^u B_u(X^p,\theta)$, where $B_u, \tilde{A}_v$, $0 \leq u, v \leq p-1$ are polynomials in $\mathbb{K}[X^p,\theta]$ of bidegree at most $(\lfloor n/p \rfloor, n)$; this costs no ops. The commutation $\theta^j X^v = X^v(\theta+v)^j$ then enables one to rewrite A as $\sum_{v=0}^{p-1} A_v(X^p,\theta)X^v$, where $A_v(X^p,\theta)$ is $\tilde{A}_v(X^p,\theta-v)$. Thus, each A_v is obtained by computing $\lfloor n/p \rfloor + 1$ shifts of polynomials of degree at most n. By Lemma 1(a), this results in $\mathcal{O}\big(n\,\mathsf{M}(n)\log n\big)$ ops. for Step 1.

Each product in Step 2 involves polynomials in $\mathbb{K}[X^p,\theta]$ of bidegree at most $(\lfloor n/p \rfloor, n)$. Thus using Lemma 1(d), Step 2 is performed in $\mathcal{O}\big(p^2\,\mathsf{M}(n^2/p)\big) \subseteq \mathcal{O}\big(p\,\mathsf{M}(n^2)\big)$ ops. Note that $C_{u,v}(X,Y)$ has bidegree at most $(2\lfloor n/p \rfloor, 2n)$.

To perform Step 3, each $C_{u,v}(X^p,\theta)X^v$ is first rewritten as $X^v \tilde{C}_{u,v}(X^p,\theta)$ by computing $2\lfloor n/p \rfloor + 1$ shifts of polynomials of degree at most $2n$. This can be done in $\mathcal{O}\big(pn\,\mathsf{M}(n)\log n\big)$ ops. Finally, $\mathcal{O}(pn^2)$ ops. are sufficient to put $C = \sum_{u=0}^{p-1} X^u \sum_{v=0}^{p-1} X^v \tilde{C}_{u,v}(X^p,\theta)$ in canonical form.

Summarizing, we have just proved:

Theorem 5 *Let $\mathbb{K}$ be a field of characteristic p and let D be one of the operators ∂, θ. Then, two operators of bidegree (n,n) in $\mathbb{K}[X]\langle D\rangle$ can be multiplied in $\mathcal{O}\big(p\,\mathsf{M}(n^2) + pn\,\mathsf{M}(n)\log n\big)$ ops., thus in $\tilde{\mathcal{O}}(pn^2)$ ops. when FFT is used.*

6. EXPERIMENTS

Table 2 provides timings of calculations in magma by implementations of several algorithms and algorithmic variants. Each row corresponds to calculations on the same pair of randomly generated operators in bidegree (n,n), for $n = 10 \cdot 2^k$. Coefficients are taken randomly from $\mathbb{Z}/p\mathbb{Z}$ when $p > 0$, the prime used being $p_1 = 65521$ (largest prime to fit on 16 bits) and $p_2 = 4294967291$ (largest prime to fit on 32 bits). When $p = 0$, computations are performed over $\mathbb{Q}$, with random integer input coefficients on 16 bits.

p	k	S	B	BZ	vdH	Iter	Tak	Rec	Int	BZI	vdHI
p_1	3	0.25	0.26	0.25	0.39	0.32	1.23	0.01	0.64	5.22	59.8
p_1	4	0.95	0.97	0.95	1.68	4.13	12.09	0.03	4.37	35.0	418
p_1	5	4.08	4.11	4.34	8.10	37.2	123	0.20	30.2	240	2793
p_1	6	21.4	21.1	22.2	45.1	397	1407	1.56	209	1692	∞
p_1	7	107	105	104	275	∞	∞	13.3	1507	∞	∞
p_2	3	0.50	0.63	0.62	1.08	2.25	5.61	0.08	1.10	8.00	82.2
p_2	4	2.24	2.66	2.68	4.52	19.07	67.73	0.35	9.22	58.2	602
p_2	5	12.2	14.5	14.1	24.4	187	926	1.63	75.6	420	∞
p_2	6	88.1	111	114	172	2604	∞	9.40	770	3146	∞
p_2	7	1961	2452	2633	∞	∞	∞	59.1	∞	∞	∞
0	3	9.93	12.0	11.3	28.4	6.99	24.3	0.07	0.93	16.9	309
0	4	128	164	164	498	118	725	0.27	6.89	204	∞
0	5	2164	2737	2725	∞	2492	∞	4.37	51.4	3172	∞

Table 2: Timings on input of bidegree $(10 \cdot 2^k, 10 \cdot 2^k)$.

The calculations were performed on a Power Mac G5 with two CPUs at 2.7 GHz, 512 kB of L2 Cache per CPU, 2.5 GB of memory, and a bus of speed 1.35 GHz. The system used was Mac OS X 10.4.10, running Magma V2.13-15. Computations killed after one hour are marked ∞.

We provide several variants of our algorithm (S, B, and BZ), as well as various others: **S:** direct call to magma's matrix multiplication in order to compute $\tilde{M}^B_{2n,3n}\tilde{M}^A_{3n,2n}$; **B and BZ:** block decomposition into $n \times n$ matrices before calling magma's matrix multiplication on, respectively, 11 block products (using Strassen's algorithm) and by 8 block products (taking the nullity of 2 blocks into account as well); **vdH:** Van der Hoeven's algorithm, as described in [11], and optimized as much as possible as the implementation S above; **Iter and Tak:** iterative formulas (2) and (5); **Rec:** magma's multiplication of a $(2n+1) \times (3n+1)$-matrix by a $(3n+1) \times (2n+1)$-matrix, that is, essentially all the linear algebra performed in variant S (in practice, almost always in the cubic regime for the objects of interest); **Int:** fully interpreted implementation of Strassen's product with cubic loop under a suitable threshold; **BZI and vdHI:** variants of the implementations BZ and vdH (with evaluation-interpolation steps improved) in which magma's product of matrices has been replaced with Int.

p		p_1	p_1	p_2	p_2	p_2	0	0	0
k		3	7	3	5	7	3	4	5
LA	$\mathcal{O}(\mathsf{MM}(n))$	4%	13%	17%	16%	39%	36%	41%	52%
PP	$\mathcal{O}(n\,\mathsf{M}(n))$	13%	25%	23%	23%	18%	36%	33%	24%
OM	$\mathcal{O}(n^2)$	38%	36%	30%	27%	11%	7%	6%	5%
IO	$\mathcal{O}(n^2)$	46%	27%	30%	33%	32%	21%	20%	19%

Table 3: Fraction of time spent in matrix product (LA), polynomial products (PP), other matrix operations (OM), and other interpreted operations (IO).

Comparing the columns Rec and, for instance, S, shows that linear algebra does not take the main part of the calculation time, although its theoretical complexity dominates. In this regard, we have been very cautious in our implementation to avoid any interpreted quadratic loops. Still, the result is that those quadratic tasks dominate the computation time. Details are given in Table 3. The conclusion is that having implemented the algorithms in an interpreted language tends to parasitize the benchmarks. For comparison sake, we have also added timings for variants BZI and vdHI that use an interpreted matrix product. They both show the growth expected in theory, as well as the ratio from 8 to 96 announced in Table 1.

7. CONCLUSIONS, FUTURE WORK

Because of space limitation, various extensions could not be covered here. More results on the complexity of non-commutative multiplication of skew polynomials will be presented in an upcoming extended version [2]. Topics like multiplication of skew polynomials with unbalanced degrees and orders, or with sparse support, will be treated there. The case of rational (instead of polynomial) coefficients will also be considered. The methods of this article extend to multiplication of more general skew polynomials, in one or several variables, including for instance q-recurrences and partial differential operators.

The constants in Table 1 are all somewhat pessimistic. Tighter bounds can be obtained by, on the one hand, relaxing the naive assumption (1), on the other hand, taking advantage of the special shapes (banded, trapezoidal, etc) of the various matrices.

We also plan to provide a lower-level implementation. Hopefully, the timings would then reflect the theoretical results even better and will be close to those of naked matrix products.

Acknowledgments. This work was supported in part by the French National Agency for Research (ANR Project “Gecko”) and the Microsoft Research-INRIA Joint Centre. We thank the three referees for their valuable comments.

8. REFERENCES

[1] D. J. Bernstein. Fast multiplication and its applications. To appear in Buhler-Stevenhagen *Algorithmic number theory*.

[2] A. Bostan, F. Chyzak, and N. Le Roux. Skew-polynomial products by evaluation and interpolation. In preparation.

[3] A. Bostan, F. Chyzak, Z. Li, and B. Salvy. Common multiples of linear ordinary differential and difference operators. In preparation.

[4] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, August 2005.

[5] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.

[6] F. Chyzak. http://algo.inria.fr/chyzak/mgfun.html.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, Mar. 1990.

[8] J. von zur Gathen and J. Gerhard. Fast algorithms for Taylor shifts and certain difference equations. In *Proceedings of ISSAC'97*, pages 40–47, New York, 1997. ACM Press.

[9] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, 1999.

[10] J. Gerhard. Modular algorithms for polynomial basis conversion and greatest factorial factorization. In *RWCA'00*, pages 125–141, 2000.

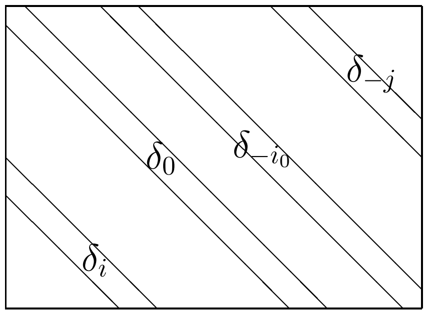
[11] J. van der Hoeven. FFT-like multiplication of linear differential operators. *Journal of Symbolic Computation*, 33(1):123–127, 2002.

[12] I. Kaporin. The aggregation and cancellation techniques as a practical tool for faster matrix multiplication. *Theor. Comput. Sci.*, 315(2-3):469–510, 2004.

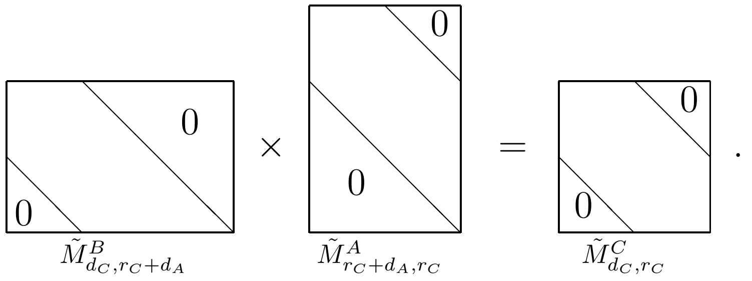
[13] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[14] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[15] N. Takayama. http://www.math.kobe-u.ac.jp/KAN/.

$\tilde{M}^{P}_{d,r} =$ 

.


0
0
0
0
0
0
$\tilde{M}^B_{d_C,r_C+d_A}$
$\tilde{M}^A_{r_C+d_A,r_C}$
$\tilde{M}^C_{d_C,r_C}$